

kind

 3 minute read  page test

`kind` is a tool for running local Kubernetes clusters using Docker container `nodes`. `kind` was primarily designed for testing Kubernetes itself, but may be used for local development or CI. Follow these instructions to prepare a `kind` cluster for Istio installation.

Prerequisites

- Please use the latest Go version.
- To use kind, you will also need to install docker.
- Install the latest version of kind.
- Increase Docker's memory limit.

Installation steps

1. Create a cluster with the following command:

```
$ kind create cluster --name istio-testing
```

--name is used to assign a specific name to the cluster. By default, the cluster will be given the name `kind`.

2. To see the list of kind clusters, use the following command:

```
$ kind get clusters  
istio-testing
```

3. To list the local Kubernetes contexts, use the following command.

```
$ kubectl config get-contexts
```

CURRENT	NAME	CLUSTER	AUTHINFO
0	NAMESPACE		
*	kind-istio-testing	kind-istio-testing	kind-istio-testing
	minikube	minikube	minikube



`kind` is prefixed to the context and cluster names, for example: `kind-istio-testing`

4. If you run multiple clusters, you need to choose which cluster `kubectl` talks to. You can set a default cluster for `kubectl` by setting the current

context in the Kubernetes kubeconfig file.

Additionally you can run following command to set the current context for `kubectl`.

```
$ kubectl config use-context kind-istio-testing  
Switched to context "kind-istio-testing".
```

Once you are done setting up a kind cluster, you can proceed to install Istio on it.

5. When you are done experimenting and you want to delete the existing cluster, use the following command:

```
$ kind delete cluster --name istio-testing  
Deleting cluster "istio-testing" ...
```

Setup Dashboard UI for kind

kind does not have a built in Dashboard UI like minikube. But you can still setup Dashboard, a web based Kubernetes UI, to view your cluster. Follow these instructions to setup Dashboard for kind.

1. To deploy Dashboard, run the following command:

```
$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.1.0/aio/deploy/recommended.yaml
```

2. Verify that Dashboard is deployed and running.

```
$ kubectl get pod -n kubernetes-dashboard
```

NAME	READY	STATUS
dashboard-metrics-scraper-76585494d8-zdb66	1/1	Running
kubernetes-dashboard-b7ffbc8cb-zl8zg	1/1	Running

3. Create a `ClusterRoleBinding` to provide admin access to the newly created cluster.

```
$ kubectl create clusterrolebinding default-admin --cluster  
role cluster-admin --serviceaccount=default:default
```

4. To login to Dashboard, you need a Bearer Token.

Use the following command to store the token in a variable.

```
$ token=$(kubectl get secrets -o jsonpath="{.items[?(@.metadata.annotations['kubernetes\.io/service-account\.name']=='default')].data.token}"|base64 --decode)
```

Display the token using the `echo` command and copy it to use for logging into Dashboard.

```
$ echo $token
```

5. You can Access Dashboard using the `kubectl` command-line tool by running the following command:


```
$ kubectl proxy
```

```
Starting to serve on 127.0.0.1:8001
```

Click [Kubernetes Dashboard](#) to view your deployments and services.



You have to save your token somewhere, otherwise you have to run step number 4 everytime you need a token to login to your Dashboard.